**Krishicare: A Novel Low-Cost Scalable Deep Learning Engine for Fast Detection and Precise Diagnosis of Infectious Diseases in Tomatoes**

**Abstract**

According to scientific reports and field results collected over years, most of the critical infections in tomatoes such as blights (early and late), mold, spot, mosaic etc. are primarily caused by bacterial and fungal Pathogens. Different diseases have different effects. Some of them affect the stem lesions, some target the leaf, some cause rots in the fruits and some can destroy crops. The remedy for each disease varies; therefore, it should be detected at an early stage. Krishicare Model is a deep learning scalable model deployed on TVM that can detect tomato diseases within a few seconds. It will help farmers take better precautions against the diseases, thereby, protecting their harvest with the help of early detection, prevention, and treatment.

Keywords: Krishicare; Tomato disease recognition; Deep learning; Tensor Virtual Machine;

**1. Introduction**

Diseases in Tomato plants are mostly caused due to weather conditions, insects and the Greenhouse effect. There are nine diseases of Tomato and a detailed analysis of each disease is as follows:

Tomato Early Blight: It is caused by the fungus *Alternaria solani*. The majority of the symptoms emerge on leaves at any stage of growth. It produces distinctive dark concentric spots on leaves and causes stem lesions and fruit rot on tomatoes. After harvest, infected plant debris in the field serves as the primary source of inoculum. It is left in the form of conidia and mycelium. In addition to this it affects other hosts like potatoes and chillies, it also spreads through weeds like *Solanum nigrum*.

Tomato Late Blight: It is caused by *Phytophthora infestans*, it is considered to be more of a fungus-like organism than a real fungus. Worldwide distribution, however, the worst outbreaks take place in regions with lots of chilly, humid weather. Solanaceous crops, such as tomato, nightshade (Solanum nigrum), and potato are the primary hosts. On tomato leaves, lesions can cover significant portions of the leaf and start as sporadic, water-soaked patches that quickly grow into pale green to brownish-black lesions. A grey to white mouldy growth may be present on the leaf's abaxial surface (not to be confused with powdery mildew disease). As the illness worsens, the leaves coil, shrivel, and eventually die after becoming yellow and then brown. Infected tomato fruits can become shrivelled, oily, olivaceous-brown, rot, and fall off the plant without ever ripening. Such diseased fruits are not suitable for human consumption.

Tomato Leaf Mold: Pathogen *Passalora fulva* causes leaf mold due to which leaves die or whither and is particularly troublesome in greenhouses and high tunnels. High tunnels extend the growing season for high-value crops like tomatoes, but they also encourage the development of tomato leaf mould, which thrives in humid, dry environments. Because tomatoes are a high-value crop and consistent yields are a significant source of income for growers, management is still essential despite the disease's slower growth than a pathogen like Phytophthora infestans.

Tomato Septoria Leaf Spot: A fungus, *Septoria lycopersici* cause septoria leaf spot. It is one of the most destructive diseases of tomato foliage and is particularly severe in wet, humid areas. After the first fruit sets, a septoria spot usually develops on the lower leaves surface. The circular spots with tan to grey centres with small black fruiting structures and dark brown borders.

Tomato Bacterial spot is caused by *Xanthomonas campestris* pv. Vesicatoria. It is another potentially deadly disease that, in extreme situations, can result in unmarketable fruit and even plant death. Can occur anywhere tomatoes are cultivated, but it is most usually found in warm, damp conditions as well as in greenhouses. Spots are small, occasionally circular patches on leaves that look moist (or water-soaked). Initial spots may be yellow-green, but as they age, they become darker and turn brownish-red. On green fruit, spots tend to be small, elevated, blister-like and may have a golden halo.

Two-spotted Spider mite: It is an occasional pest that attacks these crops during hot, dry weather, it's highly likely that you'll locate them towards the middle or end of summer. In the early stages tiny white and yellow flecks or even tiny holes on the leaves can be seen with silky webs encircling the tomato stems and leaves as their number increases. Once there is a significant infestation, one will then notice pale, powdery leaves that are entirely covered in webbing. Because spider mites feed on plant tissues, the leaves they are feeding eventually become brown. Heavy infestations of the two‑spotted spider mite produce fine webbing that may cover the entire plant.

Tomato Target Spot: It is caused by the fungal pathogen *Corynespora cassiicola.* Although similar to early blight it is common in tropical and subtropical regions. Early on, the target spot is usually misdiagnosed because its symptoms are hard to distinguish from those of bacterial spot and early blight. The name comes from the bull's eye pattern that frequently appears in lesions brought on by the illness. Water-soaked spots appear on the upper part of the leaf and when the size increases they become circular with grey to pale brown centres.

Tomato mosaic virus: ToMV is a plant pathogenic virus. It is particularly challenging to diagnose because the symptoms vary greatly depending on the type and age of the affected plant, the virus strain, and the environmental factors. All parts of the plant might get infected, and symptoms of the tomato mosaic virus can appear at any stage of growth. They frequently appear on foliage as a general mottling or mosaic appearance. When the plant is seriously harmed, the leaves may have elevated dark green areas that resemble ferns. Stunted leaves can also happen. The amount of fruit produced by infected plants may be drastically reduced, and the fruit that does set may be covered in yellow blotches and necrotic areas with a brown inside. Infection may be seen in fruit, leaves, stems, and petioles. The leaves become yellowish in some cases.

Tomato yellow leaf curl virus: Unquestionably one of the most harmful infections of tomatoes is the tomato yellow leaf curl virus, which restricts tomato production in many tropical and subtropical countries around the world. It is a DNA virus from the genus Begomovirus and the family Geminiviridae. The symptom is small leaves of the young saplings becoming yellow. This disease is usually spread via an insect named Silverleaf whitefly.

Many previous works have already been done on this topic. But they have not been used for commercial purposes. In order to get faster accurate results for commercial purposes, we proposed a model. It was first developed in Keras followed by Pytorch and then deployed on Tensor Virtual Machine(TVM).

**Previous works**

Disease prediction has been one of the most important topics in the field of modernized agriculture. For the detection of these categories, many models have been developed using Artificial Neural Networks, Support Vector Machines, Deep Learning, and Extreme Learning Machine (ELM) classifiers etc. It is also seen that 72% of Image Processing techniques are used and the rest 28% are IoT sensor-based.

A model was proposed using image processing and image classification techniques. It uses three methods contrast enhancement, segmentation and feature extraction. Finally, the classification was done using MLP. Also, a comparative analysis was drawn. However, the model was only able to detect whether the tomato leaf was healthy or not.

So to cope with the above limitation Alex Net was introduced. Alex Net, a Convolutional Neural Network with 5 Convolutional layers, of which 3 layers use max pooling has an accuracy of 98%.

Other pre-trained networks like GoogleNet and ResNet have also been used for disease prediction with an accuracy score of 97.19%.

A feasible solution was also proposed in which the pre-trained network VGG16 and AlexNet were used with 13,262 images fed into the network and the accuracy of the model was 97.49%.

To classify four types of tomato diseases a Convolutional Neural Network was developed using transfer learning as a training mechanism with deep learning CNN-based Alexnet. A total of 4932 images was used and the accuracy of the model was 95.75%.

LeNet model another pre-trained model has also been used to classify tomato diseases. The average accuracy of the model was 94%-95%.

Not only Deep Learning models have been used but Machine Learning has also been used. Using Kurtosis, skewness filters and Support Vector Machines(SVMs) accuracy of 93.75% was achieved.

A model has also been developed using Tree Classifier Model (TCM). 382 images were fed into the model and the model was able to categorize 5 types of diseases.

A popular Machine Learning Algorithm Random Forest has also been used to detect the disorder in Tomato leaves. The accuracy was 99.56%.

Unsupervised Learning Algorithm like Kmeans has also been used. However, K-means required pre-processing steps like standard scaling and Principle Component Analysis (PCA). The accuracy of the algorithm was 99.2%.

Another approach has been used to identify and classify tomato plants. For this CIE XYZ colour space analysis, color moment, histogram, and colour coherence was used. The accuracy of the model was estimated to be 87.2%.

But pre-trained networks took much time for training so a Convolutional Neural Network has been developed by Thair A. Salih, Ahmed J. Ali, and Mohammed N. Ahmed which categorizes five categories of diseases. They used 6202 images to train the model and the accuracy of the model is 96.34%.

A system containing Raspberry PI Controller along with Convolutional Neural Network was able to detect and classify tomato diseases. The accuracy of the whole system was 89%.

So we proposed a model which was first developed in Keras followed by Pytorch and then deployed on Tensor Virtual Machine(TVM).

**Gaps in the state of Art**

Although lots of research works have been done but each of them has limitations. There are some models which can only classify whether the leaves are healthy or unhealthy. They cannot recognize the diseases which is a major drawback as the remedy for each disease varies. The pre-trained models have been of great help but they take a lot of time to train. In some models, the number of images used was not sufficient. A small number of images can have very good accuracy. But if complex algorithms are used on a small dataset it can lead to the case of overfitting. Some models have an accuracy of 87-89%. Although the accuracy is good but for the prediction of diseases, an accuracy above 90% is better as according to prediction the cure for each disease is decided. For instance, if a low accuracy model detects that the plants are suffering from Late Blight but the actual disease is Early blight. The cures for these diseases are totally different so the end result might lead to huge financial loss as well. A major drawback is deep learning models take much time to train so it is economically not feasible for commercial purposes. To combat the issue many Machine Learning algorithms have been used. But these algorithms work well on small tabular datasets. Not only that, another disadvantage of using unsupervised machine learning algorithms like K-Means grouping to segment the image is that the recommended interaction is semi-automated since the client must identify the bunch that includes the unhealthy part unambiguously.

**2. Problem Statement**

Tomato diseases are caused mostly caused by Bacterial, Fungus and Virus families and the reason for the development of diseases are the Greenhouse effect, insects, excessively hot and humid weather, and excessively hot, dry weather. To prevent the disease from destroying large amounts of crops, close monitoring is required by farmers. Although many models have been developed. Some have the ability to classify a few diseases and some have been trained on small datasets. Some pre-trained models have very good accuracy for disease prediction but they have not been deployed on a scalable platform for commercial purposes because of time constraints.

**3. Proposed Solution**

**3.1. Initial Thoughts**

Krishicare is the name of our model. It is a Convolutional Neural Network Model which identifies the disease of the crop. Currently, the model handles only tomato diseases but in future, it can be used for other diseases as well. The model has been developed using two Deep learning frameworks: Keras and Pytorch.

**3.1.1. Using Keras**

The first version of the model has been developed in Keras. The model has two phases: The Image Transformation Phase and the Model Development Phase. In the first phase, we parsed through each image and used the OpenCV techniques for resizing and reading each image simultaneously storing the disease ids and path in a data frame. Then we split the images into train, test and validation. In the second phase, we defined the skeleton of the model comprising three states: entry flow, middle flow and exit flow. In each state, we defined the operations such as MaxPool, Relu, Convolutional layers etc. Finally, we visualized the results in the graphs and heatmap for further analysis.

**3.1.2. Using Pytorch**

The second version of the model has been developed in Pytorch and deployed on Tensor Virtual Machine. The model has three phases: Image Transformation Phase, Model Development and training-testing, and TVM Deployment. The image transform phase comprises resizing, normalisation, conversion to tensor and dividing into train and test sets. The second phase is building up a model and training the model using the images. The final phase is Deployment in which the model is deployed on Tensor Virtual Machine for faster accurate predictions.

**4. Method**

This section explains the working of the whole model. The whole model has been developed using Deep Learning frameworks. For the application part, we have used Python programming language and Integrated Development Environment(IDE). We also used Google Colab as a part of our research. It has inbuilt libraries like Numpy, Scikit Learn, Pandas etc. Any deep learning model has to go through a sequence of steps. The steps are Dataset Preparation, Data Preprocessing, Development of the Model and Deployment.

**4.1. Dataset Preparation**

A dataset is an integrated collection of data. Here our dataset comprises nine folders. Each folder has a category of disease images. An additional folder of healthy leaves has also been kept so that the model can differentiate between a healthy and a diseased one. The pictures are of high quality so that the model can capture all the features distinctly. The below table shows the number of images versus diseases.

| **Disease** | **Number of Images** |
| --- | --- |
| Tomato Early Blight | 1000 |
| Tomato Late Blight | 1909 |
| Tomato Leaf Mold | 952 |
| Tomato Septoria Leaf Spot | 1771 |
| Tomato Bacterial spot | 2127 |
| Two-spotted Spider mite | 1676 |
| Tomato Target Spot | 1404 |
| Tomato mosaic virus | 373 |
| Tomato yellow leaf curl virus | 5357 |
| Tomato Healthy Leaves | 1591 |

**Total number of images used are 18160**

**4.2. Data Preprocessing**

In a deep learning model, we require a large amount of data and data preprocessing plays a crucial role. It involves reading each image and its corresponding labels, resizing them, converting them to tensors and dividing them into train-test-validation datasets.

In the first version of the model, the data preprocessing features are as follows: Appending the labels and the path to the pandas data frame. The image resizing of 64 x 64 was done followed by normalisation. In the third step, we converted the parsed data to a NumPy array. Also, we divided the data into train, test and validation. The train validation and test ratio are 60:20:20.

In the second version, the data preprocessing features are as follows: The image resizing of 32 x 32 followed by normalisation and conversion of the images to tensors was done. We used the shuffle function to randomize the indices of the images. Then we divided the data into train and test images in which the ratio is 70:30 and the batch sizes for each set are varied accordingly. Finally, we segregate the labels and the image tensors using the iter function.

**4.3. Model Development Phase**

This is the second phase in which we develop our Convolutional Neural Network. The first model has been developed using Keras and the second version has been developed in Pytorch.

As seen earlier, in the Keras Version the model has three functions: Entry Flow, Middle Flow and Exit flow. The features of each of the functions are as follows:

In the entry flow, there are 5 convolutional layers including Batch normalization, 7 ReLU activation functions, 6 Separable convolutional layers including Batch normalization, and 3 Max pooling layers.

In the middle flow, we iterate 9 times to apply the functions to the tensors. The functions include 4 ReLU activation functions and 3 Separable convolutional layers including Batch normalization.

Finally, the exit flow has 3 ReLU activation functions, 4 Separable convolutional layers including Batch normalization,1 Maxpooling layer,1 Global average Dense Layer and 1 dense layer.

We feed the input to the entry flow function, then the middle flow is called 6 times and Finally, the exit flow function is called for the output. The model is optimized using the Adam optimizer and loss is calculated using Categorical cross-entropy. Lastly, we fit the training data to train the model, validation data to fit the model on batches with real-time data augmentation and use the test data to calculate the accuracy of the model

In the second version, we used Pytorch Deep Learning Framework. The features of the Pytorch Model are as follows:

We defined 2 convolutional layers, 2 max pool layers, a Relu activation function, 1 dropout function and a softmax function for the multi-class problem.

Then we used CrossEntropy loss to calculate the performance and the Adam optimizer to optimize the model. Then we use the training data to train our model and use the validation data to calculate the accuracy of our model.

**4.4. TVM Phase for the Pytorch Model**

This is the deployment phase. Here the mapping of the model to the targeted hardware is done. We choose the targeted hardware to be llvm. But other hardware like Raspberry, NVIDIA etc can also be used. Low-level relay graphs and optimisation techniques are used for the TVM. Lastly, we feed an image to the TVM and compare the prediction with our model prediction.

**4.5. Algorithm for Keras**

1) Import all the images.

2)Store the disease ids and the path names in a pandas dataframe.

3) Apply the transformations such as resizing, converting each image to tensor and

Normalisation using open cv

4)Divide the set of images into train test and validation in the ratio of 60:20:20.

5) Define the following 5 functions:

* sep\_bn (comprises of separable convolution 2d layer and Batch normalization)
* conv\_bn (comprises of convolution 2d layer and Batch normalization)
* The three functions: Entry flow, Middle flow and Exit flow which call the above two functions and other functions such as Max Pooling,

6)Optimize the model using Adam optimizer with a learning rate of 0.001 and calculate the loss of the model. Use Reduce learning rate when a metric has stopped improving.

7)Save the model at the best checkpoint.

8)Use ImageDataGenerator to flip the inputs and use the fit function to fit the training data into the model.

9)Now fit the validation data to calculate the accuracy of the model.

10)Now use test data to make predictions and use argmax function to find the predicted and actual values.

11)Now generate a heatmap to show the deviation between actual and predicted values.

12)Also plot the accuracy versus epochs for better interpretation of the results.

**4.6. Algorithm for Pytorch**

1) Import all the images.

2) Apply the transformations such as resizing, converting each image to a tensor and

normalisation.

3)Divide the whole set of images in the ratio of 70:30.

4) The model is created with the following features: 2 convolutional layers, the linear function has been applied 2 times and a dropout feature

5)Define a forward function that has the application Max\_pool2d 2 times , Relu activation function and the softmax function for a multi-class problem.

6) An object of the class is created and called

7) Then Cross entropy loss is to measure the performance of a classification model. The model is optimized using the Adam optimizer with a learning rate is 0.01

8) A for loop is used for iterating in a range of epochs

* Both train and validation losses are set to 0
* Train function is called to train the model
* In the training set, we start iterating over data and the target

I. We move the data and target tensors to GPU

II. We clear-the-gradients-of-all-optimized-variables using zero\_grad()

III. We call the model to predict the output. This is a forward pass

IV. We calculate batch loss using Crossentropy

V. We calculate backward-pass: compute-gradient-of-the-loss-wrt-model-

parameters using backward function

VI. We update the parameter at each step.

VII. Finally calculate the training loss and train accuracy at each step

* In the validation set we start iterating over data and the target

I. We move the data and target tensors to GPU

II. We call the model to predict the output. This is a forward pass

III. We calculate batch loss using Crossentropy

IV. Finally calculate the valid loss at each step

V. We calculate train loss and validation loss at each step and append them to

train\_losses list and valid\_losses list respectively.

9) We use model.eval() to disable the dropout

10) Once again we iterate over the validation set to calculate the correct predictions

and append the predictions to y\_pred.

11) Similarly we append the true values to the y\_true.

12) We calculate the confusion matrix,f1 score (taking average as a macro)and accuracy

using the two lists y\_pred and y\_true and save the model in dictionary format using model.state\_dict() and the name of the file is model.pth.

13) Finally we append epoch, batch, accuracy and f1\_score to the file named

‘torch\_acc\_f1\_score.txt’.

14) We will call the model class and load the saved dictionary of weights and model.eval() is used to evaluate the model.

15) We define an input\_data which has an RGB channel and shape of 32 x 32.

16) We use the torch.jit.trace to trace the model using the model and the input\_data as

parameters.

17) Now we load an image and resize it to 224 x 224.

18) Then we convert the image to a tensor using transforms.Resize(32),

transforms.ToTensor(),transforms.Normalize(mean=[0.485,0.456,0.406],std=[0.229,0

.224,0.225]

19) Expand the dimension using np.expand\_dims

20)Define a list shape\_list = [(input\_name, img.shape)]

21) Now we call the relay function of TVM mod, params =

relay.frontend.from\_pytorch(scripted\_model, shape\_list). Here scripted\_model is the

traced PyTorch model using jit (refer to step-18).

22) We specify the target to which the model should be mapped and clone the tophub repo.

23) Here the target is llvm.

24) We now build the relay passing the target, params and mod as parameters.

25) Import the graph\_executor from tvm.contrib.

26) The sequence of steps under graph\_executor:

* Pass the result of the relay to graph\_executor.
* Provide the input tensor that is the image in the form of n-d array
* Run the graph executor and store the output in a variable tvm\_output

27) Now we specify the list of keys and the class names of diseases.

28) Use np.argmax to get indices of the maximum element in the tvm output.

29) Now map the class name to the key. Simultaneously calculate the time for prediction

of the class disease.

30) Similarly feed the tensor to the torch model and get the prediction.

Simultaneously calculate the time for prediction of the class disease.

31) Print the results.

**5. Implementation**

Both the models have been developed in Python version 3.8.13. Google Collaboratory, Visual Studio Code and Jupyter notebook have been used as a part of our research. Both VS code and Google Collaboratory have been used to create the structure of the models and check the performance. For training the model, we have used Jupyter Notebook.

Many inbuilt packages have been used for the Keras version like Numpy is used for the computation of tensors. Pandas are used to store file names, disease ids and disease types.

The Convolutional Neural Network model has been developed using Keras. Keras is used for deep learning research and is used to make models that can be productized on smartphones.

For visualization of the results on the graph or on the heatmaps, we have used Matplotlib and Seaborn. The DateTime module records the date and time taken for the model execution completion. Pytz, another library, has been used to record places and timezone. OpenCV is a computer vision library used to read disease images and resize them.OS module is used to interact with the Operating System. It is used to access the directory of the images. Tqdm, a library, is used to make a progress bar.

The inbuilt packages used for the Pytorch model are Numpy for the computation of tensors, and Pillow is used for the image processing part. Sklearn, a machine learning library is used for the calculation of the f1 score, accuracy and confusion matrix. The whole Convolutional Neural Network model has been developed using Pytorch, an open-source Machine Learning framework used for deep learning research. Matplotlib is used to visualize the accuracy results on the graph.

TVM(Tensor Virtual Machine) is used to optimize the model and run on the targeted devices to predict the diseases. To calculate the prediction time of TVM and the model we have used the DateTime module.

The hardware used is common for both Deep learning frameworks. Since Deep Learning requires a huge amount of data and time to train, it requires GPU cores. First, we have used a local machine that comprises 8GB RAM(Random Access Memory) and 4 GB Graphics card. We trained the model on local computers. The time taken to train for 1 epoch on the local computer is approximately 0.00055 seconds. For deployment of the models, we have used Amazon Sagemaker. It is one of the most popular platforms that allow us to train, and deploy models. The instance used is Amazon EC2 P3. Amazon SageMaker comprises 8 NVIDIA T4 GPUs, 96 vCPUs, 100 Gbps networking, and 1.8 TB local NVMe-based SSD storage. There are three components: Model Artifacts, Training Code (Container) and Inference Code (Container). We uploaded a trained model file to the AWS SageMaker and placed the model in an S3 bucket and created an object for it. Finally, we configured the endpoints and used a few serverless services to trigger the endpoints from the outside world. To test our model and get back the predictions we used local clients.

**6. Analysis of Results**

We have analyzed the results of both the models and they are as follows:

From past experiments, the Krishicare model (Keras Version)

Table 1: Keras Model

| Epochs | Image Size | Batch Size | Accuracy |
| --- | --- | --- | --- |
| 50 | 64 | 16 | 0.9909572601318359 |
| 500 | 64 | 16 | 0.9953227043151855 |

Table 1 illustrates the accuracy of Keras model. For epochs 50 and batch size 16 the model achieved an accuracy of approximately 99.1%. Therefore it can be concluded that the model works far superior as compared to the previous works.

Now let us have a look at the current experiments, the Krishicare model (Pytorch version)

Table2: Pytorch Model

| Epochs | Image Size | Batch Size | Accuracy | F1 score |
| --- | --- | --- | --- | --- |
| 1 | 32 | 1 | 75.64243759178 | 0.67193208980 |
| 6 | 32 | 2 | 84.85682819383 | 0.807048369461 |
| 8 | 32 | 4 | 88.87665198239 | 0.849013418479 |
| 16 | 32 | 8 | 92.29074889867 | 0.900775955935 |
| 25 | 32 | 16 | 93.74082232011 | 0.91764121923 |
| 50 | 32 | 32 | 93.08002936857 | 0.91145742062 |
| 150 | 32 | 64 | 94.6035242291 | 0.93032 |
| 300 | 32 | 128 | 94.713656387 | 0.92964182187 |

Table 2 focuses on the accuracy and the f1 score of the model. For batch size 64, epochs 150 and image size 32 the accuracy of the model was approximately 94.6% with an f1 score of 0.93. F1 score is the major determinant of the working of the model. If the value is closer to 1 it means that the model works perfectly. Although the accuracy is not greater than the accuracies of other models, in CNN an accuracy of 90% and above is considered to be good.

Table 3: Deployment on TVM

| Category | Time is taken |
| --- | --- |
| Pytorch Model without TVM | 0.0013778209686279297 |
| Pytorch Model with TVM | 9.012222290039062e-05 |

Now we finally come to the TVM part. The model has been deployed on TVM. Here the targeted hardware used is llvm. We fed the picture of Tomato Early Blight. Both the TVM and the model predicted the correct result. Table 3 illustrates the time taken by each of the models. We can see that the time taken by the Tvm is very low as compared to the prediction by the Pytorch model. We can also draw the conclusion that the above accuracies are pretty well for a good model. Also, the TVM makes correct predictions of the disease. Thus we can conclude that a feasible solution has been found which will be very helpful in the coming future.

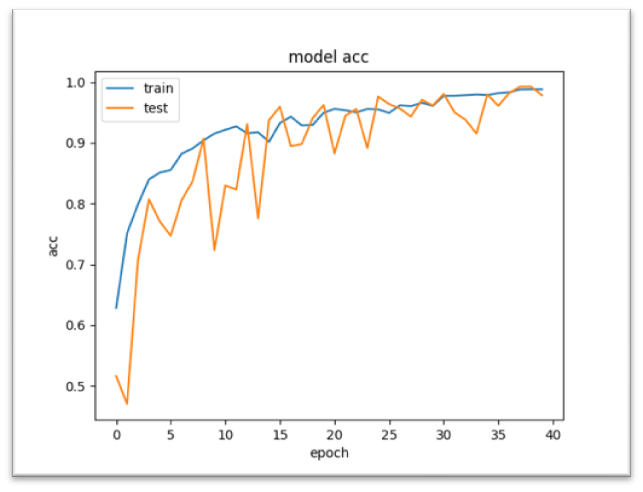
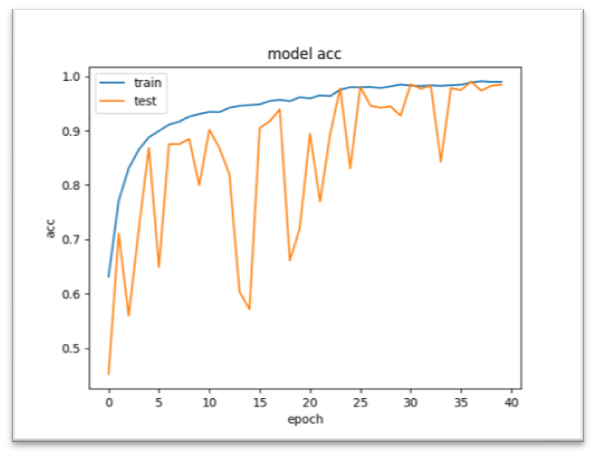
Table 4: Accuracy of both the models

| **Name of the Model** | **Accuracy(in percentage)** |
| --- | --- |
| Krishicare Keras Version | 99.09572601318359% |
| Krishicare Pytorch Version | 94.60352422907489% |

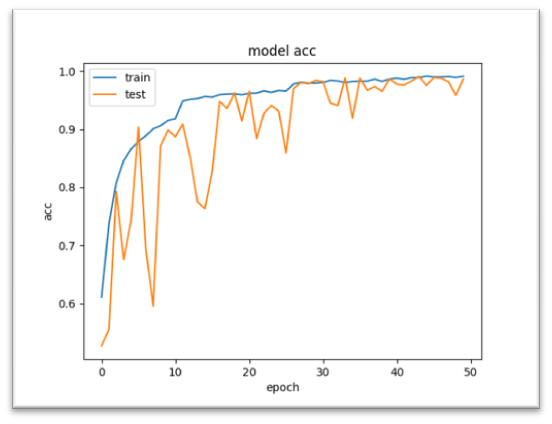
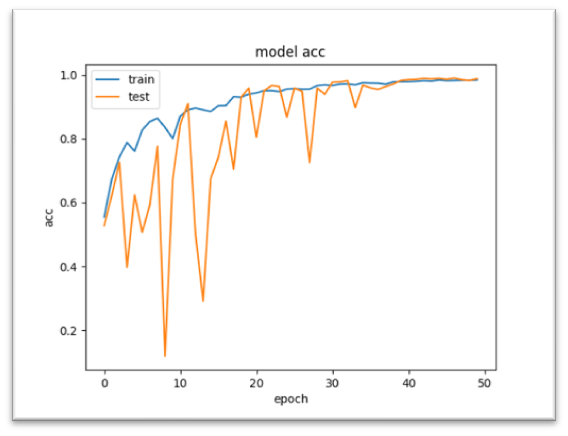
In the above table, we can see the Keras Version has a very good accuracy of 99% followed by Pytorch. However, if both the batch size and epochs are set to 1, the prediction for disease category fluctuates. But for higher epochs and batch sizes the model performs accurately. Let us have a look at the graphs for further analysis

**Keras Version**

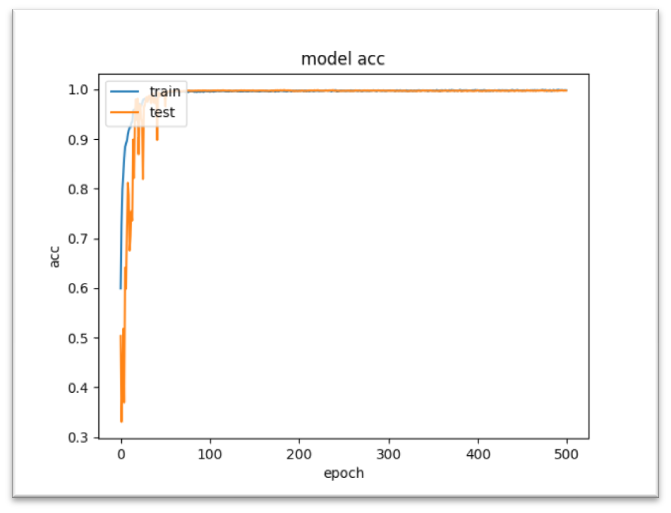
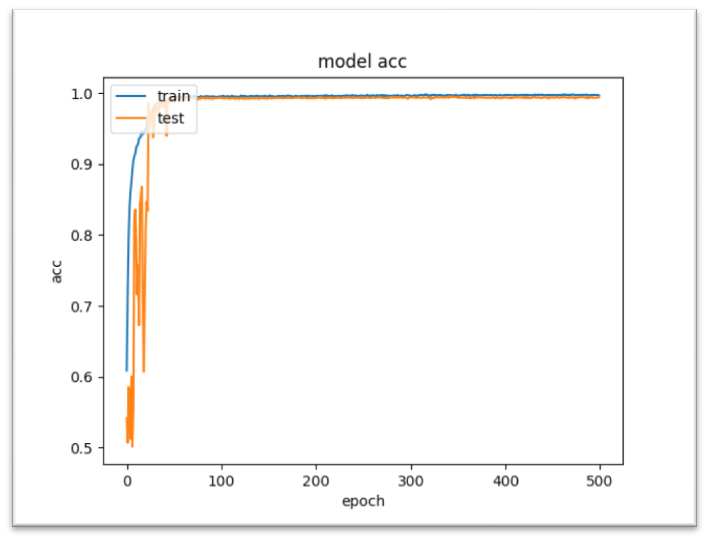
Let us have a look at the graphs. The graphs show the model accuracy versus the number of epochs.

*Figure 1: For Epochs= 40 Figure 2: For Epochs= 40*

*Figure 3: For Epochs= 50 Figure 4: For Epochs= 50*

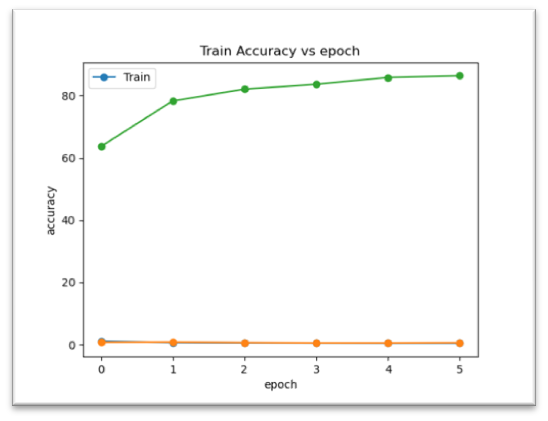
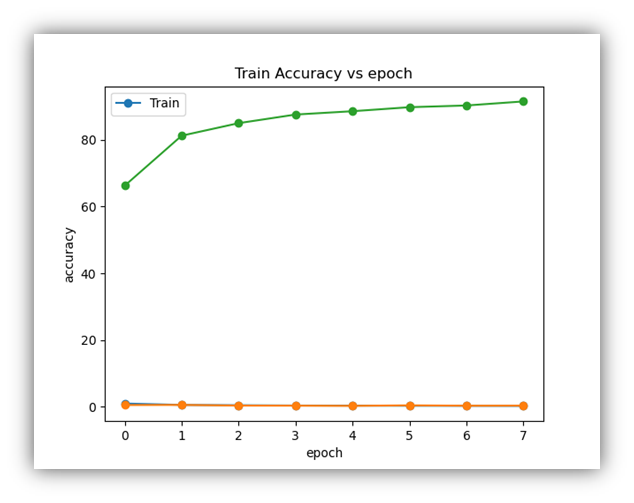
 

*Figure 5: For Epochs= 500 Figure 6: For Epochs= 500*

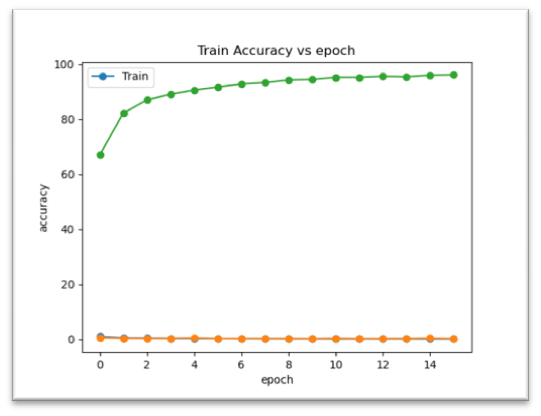
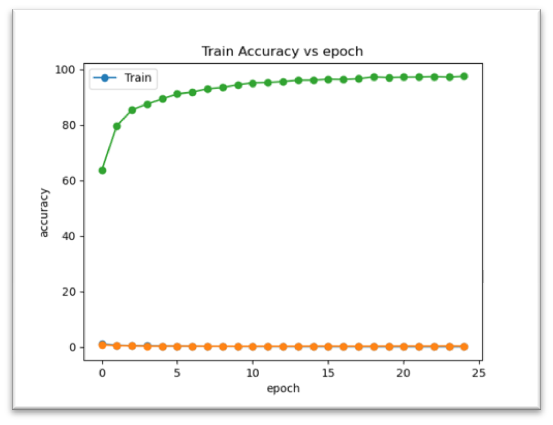
In the above graphs for lower epochs the test accuracy fluctuates but the training accuracy achieves a steady growth. For larger epochs, the test accuracy fluctuates between epochs 0 to 100 but achieves a steady growth after that. The training accuracy curve and the test accuracy curve almost merge.

**Pytorch Version**

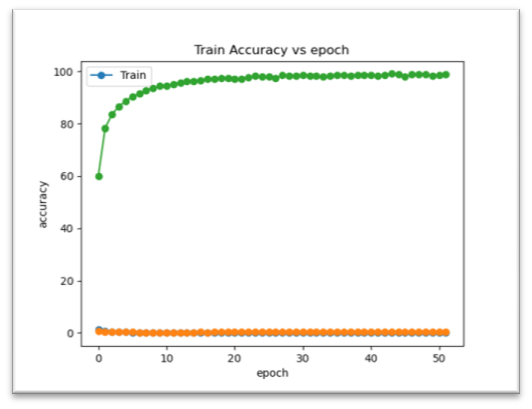
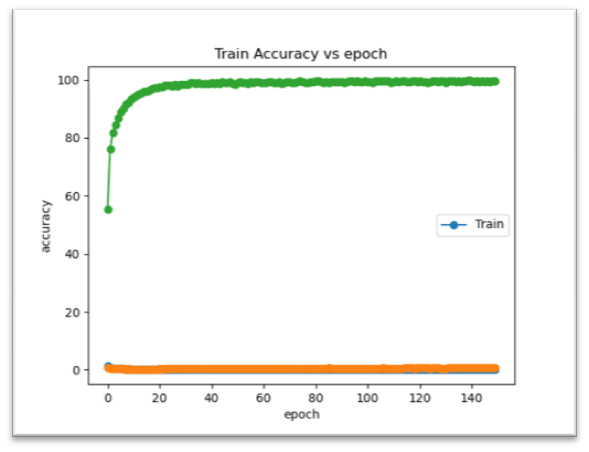
Let us have a look at the graphs. The graphs highlight the training accuracy versus the number of epochs.

** **

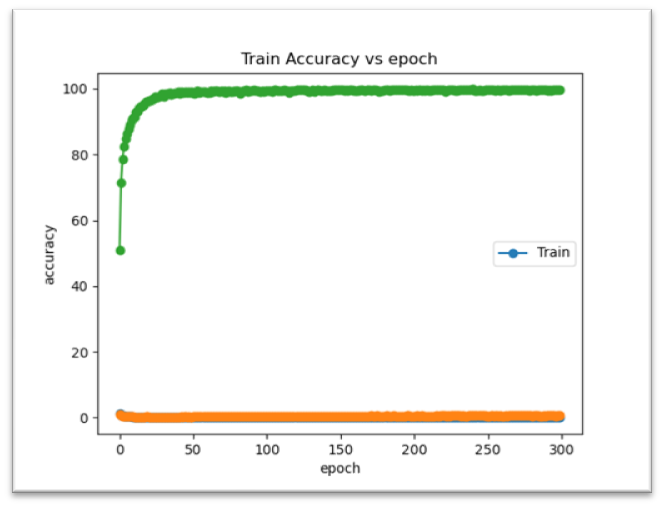
*Figure 7: For Batch size=2 epochs=6 Figure 8: For Batch size=4 epochs=8*

*Figure 9: For Batch size=8 epochs=16 Figure 10: For Batch size=16 epochs=25*

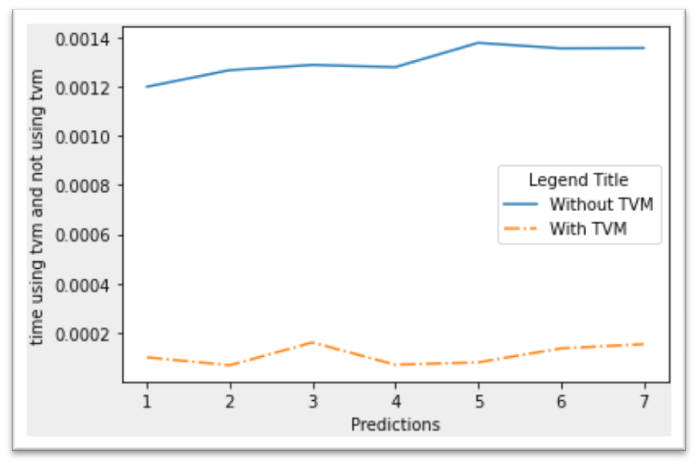
 

*Figure 11: For Batch size=32 epochs=50 Figure 12: For Batch size=64 epochs=150*

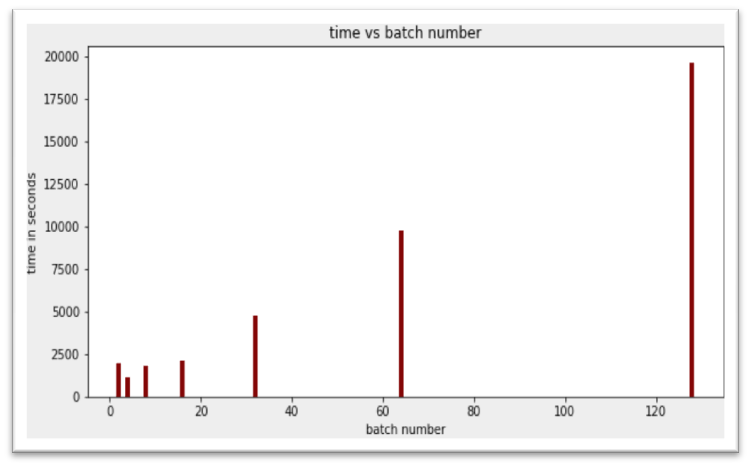
*Figure 13: For batch size = 128 epochs=300 Figure 14: Tomato Early Blight*

For the above graphs the training accuracy increases with the increase in epoch size. For larger batch and epoch sizes, the training accuracy almost remains constant. A slight kink can be seen in training accuracy for epoch size 50 but overall it has a steady growth. In figure 14 a test image of Tomato Early Blight has been used in the deployment phase.



*Figure 15: Time taken by the model to make predictions versus time taken by TVM*

From figure 15 we can see that the model prediction takes a significantly greater amount of time than TVM. The time taken by TVM is almost less than 0.0001 seconds.

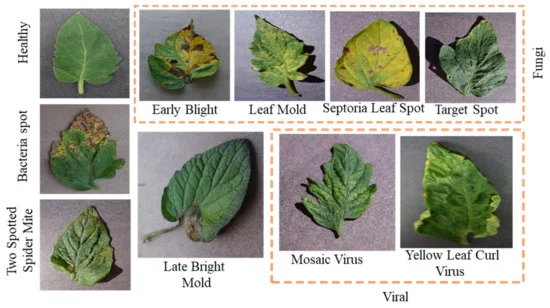


*Figure 16: Batch Number versus time taken by the model*

In figure 16, we can see a bar chart. The bar chart shows the relation between batch numbers and the time taken for the execution of the model. It can be seen with the increase in batch number also results in increased time.

**7. Confidence Score**

The model has been trained on many epochs. Both the model and the TVM make correct predictions about whether the leaf is healthy or not. It can also make correct predictions in the cases of diseases such as Early Blight, Late Blight, Two Spotted Spider Mite, and Mosaic Virus. For other diseases, the results fluctuate a little bit but if the model is trained for a long period of time it almost provides good results.

****

*Figure 17: Different Categories of Tomato Leaf*

In the above figure we can see the classification of diseases. The fungi family that causes the diseases to have almost similar characteristics yellowish leaves, spots etc. The bacterial family causes large black spots whereas viruses cause the leaves to wither and become yellowish.

**8. Ambiguous Cases**

Ambiguous results are seen in the cases of Early blight and Late Blight for very lower epochs and lower batch sizes. Sometimes the model also shows ambiguity between Bacterial Spot and Blight. There are two reasons. Firstly, the biological reason is a collection of spots result in Blight. The second reason is that these ambiguous cases arise for very low epochs and low batch sizes. Sometimes the model is unable to detect the target spot because the target spot on tomato fruit is difficult to recognize in the early stages, as the disease resembles several other fungal diseases of tomatoes. Some diseases have similar features. For example, Bacterial Spot also results in yellowish spots and leaf mold also results in yellowish spots. On the other hand, Septoria leaf spot and Leaf mold have similar features. However, if we use more images for training and finally tune the epochs and the batch size it will give the expected results.

**9. Impact**

Agriculture is the most important sector and serves as the base for other sectors. We all know that growing crops involves a huge amount of money as well as proper care. So if proper care is not taken it can lead to huge loss. Also, it is not feasible for farmers to monitor each plant. But there is one solution that can surely come to farmers' rescue and that is Deep Learning. With the help of this technology, we can detect diseases of the plant and prevent them at an early stage.

Krishicare Model helps to detect diseases. Although it works only for Tomato crops, this model has good learning and feature extraction ability. To detect the category of the disease, the farmer only needs to feed some images of the leaf and it will predict the category within a few seconds, thanks to the capability of TVM. So we can conclude the model is feasible and can be helpful for the agriculture sector.

**10. Discussion**

The models have very good accuracies. But the best choice is the Krishicare model developed in Pytorch Deep Learning framework with epochs =100 or 50, batch size = 32 deployed on TVM. Although Keras has a significant edge over Pytorch in terms of accuracy, if we see it from the practical point of view TVM provides faster accurate results. The TVM allows the deployment of the model on any hardware, therefore making it much more feasible and easier for commercial use. The main advantage of our model over other models is that our model has been trained on large image datasets and it is neither overfitting nor underfitting. Besides the TVM also provides accurate results. Our model is not very complex. These are the features which provide an upper hand over other models.

**11. Future work**

Both the models have good accuracy, but definitely, some improvements can be made. More images can be used for training the model so that the model can also cope with ambiguity. Since the diseases are caused by the Fungal, Bacterial and Virus families, the diseases have similar characteristics as well. The model should capture more features so that it can make accurate predictions. The expansion should be done so that the models can make accurate predictions of other crop diseases as well. If the models are deployed, they will be useful for farmers. An addition can be made that besides predicting the disease, the models can provide remedies for each disease. This will surely help in the whole development of agriculture.

**12. Conclusion**

In this work we successfully developed a model using two Deep Learning Frameworks. The model is able to classify whether the tomato leaf is healthy or not and also categorises which type of disease it is. Both models have accuracy above 90%. To make the predictions faster, the model has been deployed on TVM which is able to make predictions in fewer seconds.

Agriculture is the backbone of any country. Any disease if not controlled can cause millions of financial losses and can ultimately hamper GDP. However, if we use the latest technologies to combat these problems, it will be a huge boon for any country. It can help farmers immediately to identify diseases and provide proper remedies such as using proper pesticides, fertilizers etc. Thus agriculture can be modernized by using these latest technologies.

**References**

[1]Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network Naresh K. Trivedi , Vinay Gautam , Abhineet Anand, Hani Moaiteq Aljahdali , Santos Gracia Villar , Divya Anand , Nitin Goyal and Seifedine Kadry Network. Sensors 2021, 21, 7987. <https://doi.org/10.3390/s21237987>

[2]Tomato plant disease classification in digital images using classification tree [H. Sabrol](https://ieeexplore.ieee.org/author/37086198839),

[K. Satish](https://ieeexplore.ieee.org/author/37086204668) IEEE <https://ieeexplore.ieee.org/document/7754351> 24 November 2016 DOI**:** [10.1109/ICCSP.2016.7754351](https://doi.org/10.1109/ICCSP.2016.7754351)

[3]Tomato crop disease classification using pre-trained deep learning algorithm Aravind Krishnaswamy Rangarajan, Raja Purushothaman, Aniirudh Ramesh [Procedia Computer Scien](https://www.sciencedirect.com/journal/procedia-computer-science)ce [Volume 133](https://www.sciencedirect.com/journal/procedia-computer-science/vol/133/suppl/C), 2018, Pages 1040-1047 <https://doi.org/10.1016/j.procs.2018.07.070>

[4]Tomato Diseases & Disorders Factsheet | HGIC 2217 | Updated: May 25, 2021

[5]Abrahamian, P., Klein, J. M., Jones, J. B., Vallad, G. E., and Melanson, R. A. 2019. First report of bacterial spot of tomato caused by Xanthomonas perforans in Mississippi. Plant Disease 103(1):147

[6]AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf Hsing-Chung Chen , Agung Mulyo Widodo , Andika Wisnujati , Mosiur Rahaman , Jerry Chun-Wei Lin, Liukui Chen and Chien-Erh Weng Published: 18 March 2022 Electronics 2022, 11, 951. <https://doi.org/10.3390/electronics11060951>

[7]Disease management of tomato through PGPB: current trends and future perspective [Vipin Kumar Singh](https://pubmed.ncbi.nlm.nih.gov/?term=Singh%20VK%5BAuthor%5D), [Amit Kishore Singh](https://pubmed.ncbi.nlm.nih.gov/?term=Singh%20AK%5BAuthor%5D), and [Ajay Kumar](https://pubmed.ncbi.nlm.nih.gov/?term=Kumar%20A%5BAuthor%5D) National Library of Medicine Published online 2017 Jul 20. doi: [10.1007/s13205-017-0896-1](https://doi.org/10.1007%2Fs13205-017-0896-1)

[8]IMAGE CLASSIFICATION USING PYTORCH ABHISHEK PANDEY International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

[9]Deep Learning Techniques in Tomato Plant – A Review S Mohana Saranya, R R Rajalaxmi , R Prabavathi , T.Suganya , S.Mohanapriya , K.Tamilselvi IOP Publishing doi:10.1088/1742-6596/1767/1/012010

[10]Tomato Leaf Diseases Detection Using Deep Learning Technique Muhammad E.H. Chowdhury, Tawsifur Rahman, Amith Khandakar, Nabil Ibtehaz, Aftab Ullah Khan, Muhammad Salman Khan, Nasser Al-Emadi, Mamun Bin Ibne Reaz, Mohammad Tariqul Islam and Sawal Hamid Md. Ali Published: April 30th, 2021 DOI: 10.5772/intechopen.97319

[11]https://www.mdpi.com/2624-7402/3/2/20/htm